10. Given a File of N employee records with a set K of Keys(4-digit) which

uniquely determine the records in file F.

Assume that file F is maintained in memory by a Hash Table (HT) of m

memory locations with L as the set of memory addresses (2-digit) of

locations in HT.

Let the keys in K and addresses in L are integers.

Design and develop a Program in C that uses Hash function H: K -&gt; L as

H(K)=K mod m (remainder method), and implement hashing technique to

map a given key K to the address space L.

Resolve the collision (if any) using linear probing.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 100

#define M 10

typedef struct {

int key,

char name[50];

char department[30];

} Employee;

typedef struct {

int key;

Employee emp;

int isOccupied;

} HashTableEntry;

int hashFunction(int key) {

return key % M;

}

void insert(HashTableEntry hashTable[], Employee emp) {

int index = hashFunction(emp.key);

int originalIndex = index;

while (hashTable[index].isOccupied) {

if (hashTable[index].key == emp.key) {

printf("Error: Duplicate key detected!\n");

return;

}

index = (index + 1) % M;

if (index == originalIndex) {

printf("Error: Hash table is full!\n");

return;

}

}

hashTable[index].key = emp.key;

hashTable[index].emp = emp;

hashTable[index].isOccupied = 1;

printf("Inserted key %d at index %d\n", emp.key, index);

}

Employee \*search(HashTableEntry hashTable[], int key) {

int index = hashFunction(key);

int originalIndex = index;

while (hashTable[index].isOccupied) {

if (hashTable[index].key == key) {

return &hashTable[index].emp;

}

index = (index + 1) % M;

if (index == originalIndex) {

break;

}

}

return NULL;

}

void displayHashTable(HashTableEntry hashTable[]) {

printf("\nHash Table:\n");

for (int i = 0; i < M; i++) {

if (hashTable[i].isOccupied) {

printf("Index %d: Key = %d, Name = %s, Department = %s\n",

i, hashTable[i].key, hashTable[i].emp.name, hashTable[i].emp.department);

} else {

printf("Index %d: Empty\n", i);

}

}

}

int main() {

HashTableEntry hashTable[M];

for (int i = 0; i < M; i++) {

hashTable[i].isOccupied = 0;

}

int choice;

Employee emp;

do {

printf("\nMenu:\n");

printf("1. Insert Employee Record\n");

printf("2. Search Employee Record\n");

printf("3. Display Hash Table\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter 4-digit key: ");

scanf("%d", &emp.key);

printf("Enter name: ");

scanf("%s", emp.name);

printf("Enter department: ");

scanf("%s", emp.department);

insert(hashTable, emp);

break;

case 2:

printf("Enter key to search: ");

int key;

scanf("%d", &key);

Employee \*result = search(hashTable, key);

if (result) {

printf("Employee Found: Key = %d, Name = %s, Department = %s\n",

result->key, result->name, result->department);

} else {

printf("Employee with key %d not found.\n", key);

}

break;

case 3:

displayHashTable(hashTable);

break;

case 4:

printf("Exiting...\n");

break;

default:

printf("Invalid choice!\n");

}

} while (choice != 4);

return 0;

}





